

Symfony 5 : contrôleur

Achref El Mouelhi

Docteur de l'université d'Aix-Marseille
Chercheur en Programmation par contrainte (IA)
Ingénieur en Génie logiciel

`elmouelhi.achref@gmail.com`



- 1 Introduction
- 2 Génération d'un contrôleur
- 3 Routage
 - Routage par attribut
 - Routage par annotation
 - Routage dans un fichier YAML
 - Routage dans un fichier PHP
 - Routage dans un fichier XML
- 4 Multi-routes

5 Paramètres de substitution

- Validation de paramètres
- Paramètres optionnels
- Paramètres avec `null` comme valeur par défaut
- Paramètres supplémentaires
- Priorité

6 Débogage des routes

7 Objet `request`

- Récupération des paramètres de substitution
- Récupération des paramètres hors route
- Récupération de nom de la route
- Autres informations

Plan

- 8 Méthode HTTP
- 9 Expression Language
- 10 Génération d'URL et redirection
 - `forward`
 - `generateUrl`
 - `redirect`
 - `redirectToRoute`
- 11 Gestion d'erreurs et page 404
- 12 Objet `response`
- 13 Paramètres de l'application
- 14 Bonnes pratiques

Symfony

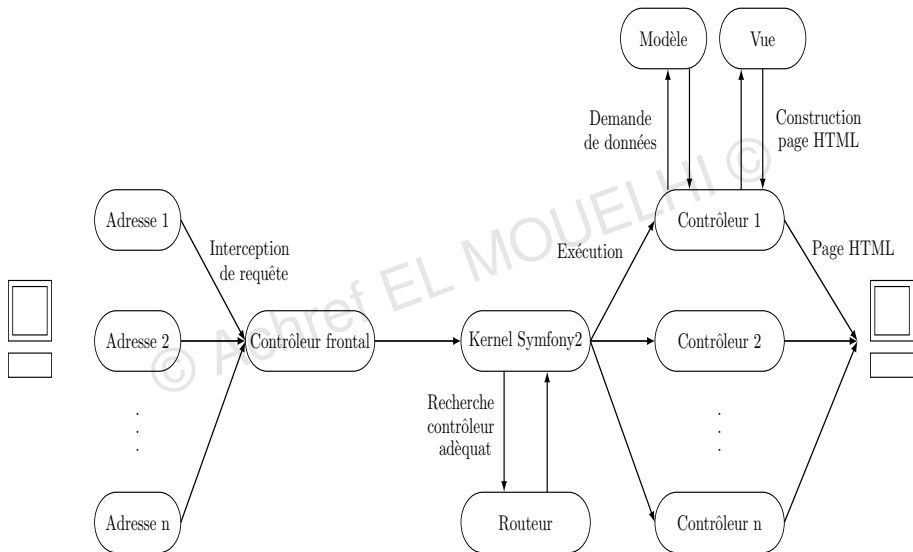
Rôle

- Élément indispensable de l'architecture **MVC**
- Il reçoit une requête et il interagit avec les différents composants d'une application **Symfony** :
 - les vues
 - les services
 - les modèles
 - les constructeurs de formulaires
 - ...
- pour retourner une réponse

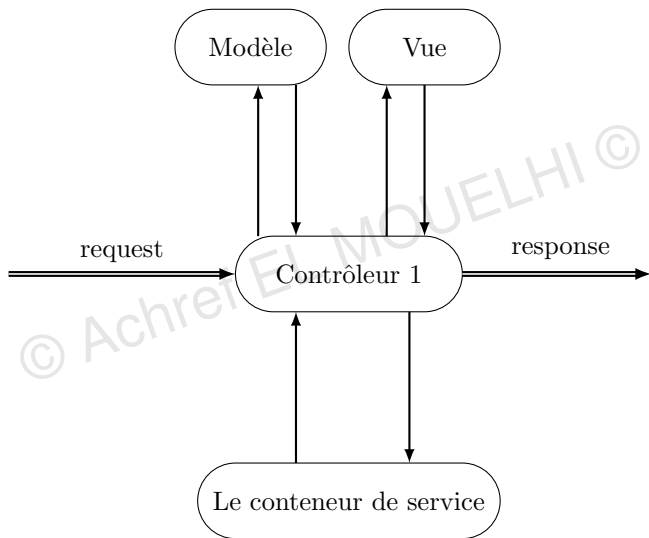
Techniquement

- Un contrôleur est une classe **PHP** qui hérite d'`AbstractController`
- Une méthode (action) de contrôleur est associée à une ou plusieurs routes
- Dans un contrôleur, il n'y a que du code **PHP** (pas de **HTML** ni **CSS** ni **JS**)

Symfony



Symfony



Symfony

Explication

- `request` **et** `response` sont deux objets
- `request` contient les données concernant la requête utilisateur
- `response` correspond à la réponse préparée puis retourner par le contrôleur
- Les services, les modèles... vont nous permettre de réaliser tout le travail nécessaire pour préparer le contenu de la réponse.

Symfony

Pour générer un contrôleur nommé HomeController

```
php bin/console make:controller HomeController
```

© Achref EL MOUELHI ©

Symfony

Pour générer un contrôleur nommé HomeController

```
php bin/console make:controller HomeController
```

Le résultat est

```
created: src/Controller/HomeController.php  
created: templates/home/index.html.twig
```

Symfony

Pour générer un contrôleur nommé `HomeController`

```
php bin/console make:controller HomeController
```

Le résultat est

```
created: src/Controller/HomeController.php  
created: templates/home/index.html.twig
```

Constats

- `HomeController.php` : un contrôleur généré dans `src/controller`
- `index.html.twig` : une vue générée dans `templates/home`
- `home` : un répertoire créé pour le contrôleur `HomeController` qui contiendra toutes ses vues. Par défaut, **Symfony** cherchera les vues dans ce répertoire.

Pour générer un contrôleur sans template

```
php bin/console make:controller HomeController --no-template
```

© Achref EL MOUËL

Symfony

Pour générer un contrôleur sans template

```
php bin/console make:controller HomeController --no-template
```

Si on oublie de spécifier le nom, Symfony nous le rappellera

```
Choose a name for your controller class (e.g.  
GrumpyPizzaController):
```

Symfony

Plusieurs modes de routage avec **Symfony**

- par attribut (par défaut depuis la version 5.3)
- par annotation (par défaut avant la version 5.3)
- dans un fichier **YAML**
- dans un fichier **XML**
- dans un fichier **PHP**

Symfony

Plusieurs modes de routage avec **Symfony**

- par attribut (par défaut depuis la version 5.3)
- par annotation (par défaut avant la version 5.3)
- dans un fichier **YAML**
- dans un fichier **XML**
- dans un fichier **PHP**

Pour le vérifier, allez dans `config/routes/annotations.yaml`

```
controllers:  
    resource: ../../src/Controller/  
    type: annotation
```


Symfony

Code généré pour HomeController

```
namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Routing\Annotation\Route;

class HomeController extends AbstractController
{
    #[Route('/home', name: 'home')]
    public function index(): Response
    {
        return $this->render('home/index.html.twig', [
            'controller_name' => 'HomeController',
        ]);
    }
}
```

Symfony

Explication

- Tous les contrôleurs sont définis dans un namespace `App\Controller`.
- La méthode `index` retourne la vue `home/index.html.twig` et lui envoie un paramètre `controller_name` avec comme valeur le nom du contrôleur `HomeController`.
- La méthode `index` est précédée par `#[Route]` qui définit le chemin qui permettra d'exécuter cette méthode.
- L'attribut `#[Route]` permet d'associer un nom à la route pour qu'on puisse l'appeler.

Symfony

Explication

- Tous les contrôleurs sont définis dans un namespace `App\Controller`.
- La méthode `index` retourne la vue `home/index.html.twig` et lui envoie un paramètre `controller_name` avec comme valeur le nom du contrôleur `HomeController`.
- La méthode `index` est précédée par `#[Route]` qui définit le chemin qui permettra d'exécuter cette méthode.
- L'attribut `#[Route]` permet d'associer un nom à la route pour qu'on puisse l'appeler.

Pour tester, allez à `localhost:8000/home`

Symfony

Code généré pour HomeController

```
namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Routing\Annotation\Route;

class HomeController extends AbstractController
{
    /**
     * @Route("/home", name="home_route")
     */
    public function index(): Response
    {
        return $this->render('home/index.html.twig', [
            'controller_name' => 'HomeController',
        ]);
    }
}
```

Symfony

Explication

- La méthode `index` est annotée par `@Route` qui définit le chemin qui permettra d'exécuter cette méthode.
- L'annotation `@Route` permet d'associer un nom à la route pour qu'on puisse l'appeler.

© Acti

Symfony

Explication

- La méthode `index` est annotée par `@Route` qui définit le chemin qui permettra d'exécuter cette méthode.
- L'annotation `@Route` permet d'associer un nom à la route pour qu'on puisse l'appeler.

Pour tester, allez à `localhost:8000/home`

Symfony

Routage dans un fichier **YAML** : démarche

- supprimer l'annotation de la méthode `index` du contrôleur `HomeController`
- vérifier que la route `/home` n'est plus accessible
- commenter la partie concernant l'annotation des contrôleurs dans `annotations.yaml`
- définir les routes dans `routes.yaml`

Symfony

Nouveau contenu de HomeController

```
namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Routing\Annotation\Route;

class HomeController extends AbstractController
{
    public function index(): Response
    {
        return $this->render('home/index.html.twig', [
            'controller_name' => 'HomeController',
        ]);
    }
}
```


Symfony

Nouveau contenu de annotations.yaml

```
# controllers:
#     resource: ../../src/Controller/
#     type: annotation

kernel:
    resource: ../../src/Kernel.php
    type: annotation
```

Symfony

Définissons les routes dans `routes.yaml`

```
home:  
  path: /home  
  controller: App\Controller\HomeController::index
```

© Achref EL M...

Symfony

Définissons les routes dans `routes.yaml`

```
home:  
  path: /home  
  controller: App\Controller\HomeController::index
```

Pour tester, allez à `localhost:8000/home`

Routage dans un fichier **PHP** : démarche

- commenter code de `routes.yaml`
- vérifier que la route `/home` n'est plus accessible
- créer un fichier `routes.php` dans `config`
- définir les routes dans `routes.php`

Symfony

Définissons les routes dans `routes.php`

```
<?php

use App\Controller\HomeController;
use Symfony\Component\Routing\Loader\Configurator\
    RoutingConfigurator;

return function (RoutingConfigurator $routes) {
    $routes->add('home', '/home')
        ->controller([HomeController::class, 'index']);

};

?>
```

Symfony

Depuis Symfony 5.1, pour utiliser le routage XML ou PHP, il faut modifier la méthode `configureRoutes` de `src/Kernel.php`

```
protected function configureRoutes(RoutingConfigurator $routes): void
{
    $routes->import('../config/{routes}/' . $this->environment . '/*.yaml');
    $routes->import('../config/{routes}/*.yaml');

    if (is_file(\dirname(__DIR__) . '/config/routes.yaml')) {
        $routes->import('../config/routes.yaml');
    } elseif (is_file($path = \dirname(__DIR__) . '/config/routes.php')) {
        (require $path)($routes->withPath($path), $this);
    }
}
```

© Achref EL M.

Symfony

Depuis Symfony 5.1, pour utiliser le routage XML ou PHP, il faut modifier la méthode `configureRoutes` de `src/Kernel.php`

```
protected function configureRoutes(RoutingConfigurator $routes): void
{
    $routes->import('../config/{routes}/' . $this->environment . '/*.yaml');
    $routes->import('../config/{routes}/*.yaml');

    if (is_file(\dirname(__DIR__) . '/config/routes.yaml')) {
        $routes->import('../config/routes.yaml');
    } elseif (is_file($path = \dirname(__DIR__) . '/config/routes.php')) {
        (require $path)($routes->withPath($path), $this);
    }
}
```

Le nouveau code (mentionné dans le commit suivant :

<https://github.com/symfony/recipes/commit/5d24d659fc54e2ed564ae93f2f09a2fefe0c850f>)

```
protected function configureRoutes(RoutingConfigurator $routes): void
{
    $extensions = '{php,yaml,xml}';

    $routes->import('../config/{routes}/' . $this->environment . "/*.{$extensions}");
    $routes->import("../config/{routes}/*.{$extensions}");
    $routes->import("../config/{routes}.$extensions");
}
```

Symfony

Pour tester, allez à `localhost:8000/home`

© Achref EL MACHHI ©

Routage dans un fichier **XML** : démarche

- supprimer le fichier `routes.php`
- vérifier que la route `/home` n'est plus accessible
- créer un fichier `routes.xml` dans `config`
- définir les routes dans `routes.xml`

Symfony

Définissons les routes dans `routes.xml`

```
<?xml version="1.0" encoding="UTF-8" ?>
<routes xmlns="http://symfony.com/schema/routing"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://symfony.com/schema/routing
    http://symfony.com/schema/routing/routing-1.0.xsd">

  <route id="home" path="/home"
    controller="App\Controller\HomeController::index"/>

</routes>
```

Symfony

Définissons les routes dans `routes.xml`

```
<?xml version="1.0" encoding="UTF-8" ?>
<routes xmlns="http://symfony.com/schema/routing"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://symfony.com/schema/routing
    https://symfony.com/schema/routing/routing-1.0.xsd">

  <route id="home" path="/home"
    controller="App\Controller\HomeController::index"/>

</routes>
```

Pour tester, allez à `localhost:8000/home`

Symfony

Pour la suite

- Nous n'utiliserons que le routage par attribut ou par annotation.
- Pensez à supprimer tous les fichiers `routes.*`

© Achref EL MOUELHI

Symfony

Pour la suite

- Nous n'utiliserons que le routage par attribut ou par annotation.
- Pensez à supprimer tous les fichiers `routes.*`

Pensez à réactiver le routage par annotation ou attribut dans
`annotations.yaml`

controllers:

```
resource: ../../src/Controller/  
type: annotation
```

kernel:

```
resource: ../../src/Kernel.php  
type: annotation
```

Il est possible d'associer une route à un contrôleur et une deuxième à la méthode `index`. Cette dernière sera exécutée en allant à `localhost:8000/home/index`

```
namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\Routing\Annotation\Route;

/**
 * @Route("/home")
 */
class HomeController extends AbstractController
{
    /**
     * @Route("/index")
     */
    public function index(): Response
    {
        return $this->render('home/index.html.twig', [
            'controller_name' => 'HomeController',
        ]);
    }
}
```

Symfony

Ou en utilisant les attributs

```
namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Routing\Annotation\Route;

#[Route('/')]
class HomeController extends AbstractController
{
    #[Route('/index')]
    public function index(): Response
    {
        return $this->render('home/index.html.twig', [
            'controller_name' => 'HomeController',
        ]);
    }
}
```

Il est aussi possible d'associer plusieurs routes à notre méthode

```
namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\
    AbstractController;
use Symfony\Component\Routing\Annotation\Route;

class HomeController extends AbstractController
{
    /**
     * @Route("/")
     * @Route("/home", name="home_route")
     */
    public function index()
    {
        return $this->render('home/index.html.twig', [
            'controller_name' => 'HomeController',
        ]);
    }
}
```


Symfony

Pour tester, allez à `localhost:8000/home` ou `localhost:8000`

© Achref EL M. ELHI ©

Deux types de paramètre

- Paramètre de substitution : `/home/wick/john` (à définir dans la route avec `{nomParametre}`)
- Paramètre hors route : `/home?nom=wick&prenom=john` (à ne pas inclure dans la route)

Pour récupérer un paramètre de substitution, il faut le définir dans la route et l'ajouter comme paramètre de l'action

```
namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\Routing\Annotation\Route;

class HomeController extends AbstractController
{
    /**
     * @Route("/home/{nom}", name="home_route")
     */
    public function index(string $nom)
    {
        return $this->render('home/index.html.twig', [
            'controller_name' => $nom,
        ]);
    }
}
```

Pour récupérer un paramètre de substitution, il faut le définir dans la route et l'ajouter comme paramètre de l'action

```
namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\Routing\Annotation\Route;

class HomeController extends AbstractController
{
    /**
     * @Route("/home/{nom}", name="home_route")
     */
    public function index(string $nom)
    {
        return $this->render('home/index.html.twig', [
            'controller_name' => $nom,
        ]);
    }
}
```

Pour tester, allez à localhost:8000/home/wick

Symfony

Solution avec les attributs

```
namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Routing\Annotation\Route;

class HomeController extends AbstractController
{
    #[Route('/home/{nom}', name: "home_route")]
    public function index(string $nom): Response
    {
        return $this->render('home/index.html.twig', [
            'controller_name' => $nom,
        ]);
    }
}
```

Symfony

Solution avec les attributs

```
namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Routing\Annotation\Route;

class HomeController extends AbstractController
{
    #[Route('/home/{nom}', name: "home_route")]
    public function index(string $nom): Response
    {
        return $this->render('home/index.html.twig', [
            'controller_name' => $nom,
        ]);
    }
}
```

Pour tester, allez à `localhost:8000/home/wick`

On peut utiliser les expressions régulières pour définir des contraintes sur les paramètres

```
namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\Routing\Annotation\Route;

class HomeController extends AbstractController
{
    /**
     * @Route("/home/{age}", name="home_route", requirements={"age"="\d{2,3}"})
     */
    public function index(int $age)
    {
        return $this->render('home/index.html.twig', [
            'controller_name' => $age,
        ]);
    }
}
```

On peut utiliser les expressions régulières pour définir des contraintes sur les paramètres

```
namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\Routing\Annotation\Route;

class HomeController extends AbstractController
{
    /**
     * @Route("/home/{age}", name="home_route", requirements={"age"="\d{2,3}"})
     */
    public function index(int $age)
    {
        return $this->render('home/index.html.twig', [
            'controller_name' => $age,
        ]);
    }
}
```

Pour tester, allez à `localhost:8000/home/50`

Symfony

Ou en utilisant les attributs

```
namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\Routing\Annotation\Route;

class HomeController extends AbstractController
{
    #[Route('/home/{age}', name: "home_route", requirements: ["age" =>
        "\d{2,3}"))]
    public function index(int $age)
    {
        return $this->render('home/index.html.twig', [
            'controller_name' => $age,
        ]);
    }
}
```

Symfony

Ou en utilisant les attributs

```
namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\Routing\Annotation\Route;

class HomeController extends AbstractController
{
    #[Route('/home/{age}', name: "home_route", requirements: ["age" =>
        "\d{2,3}"))]
    public function index(int $age)
    {
        return $this->render('home/index.html.twig', [
            'controller_name' => $age,
        ]);
    }
}
```

Pour tester, allez à `localhost:8000/home/50`

La contrainte peut être collée au paramètre (sans l'attribut `requirements`)

```
namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\Routing\Annotation\Route;

class HomeController extends AbstractController
{
    /**
     * @Route("/home/{age<\d+>}", name="home_route")
     */
    public function index(int $age)
    {
        return $this->render('home/index.html.twig', [
            'controller_name' => $age,
        ]);
    }
}
```

La contrainte peut être collée au paramètre (sans l'attribut `requirements`)

```
namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\Routing\Annotation\Route;

class HomeController extends AbstractController
{
    /**
     * @Route("/home/{age<\d+}", name="home_route")
     */
    public function index(int $age)
    {
        return $this->render('home/index.html.twig', [
            'controller_name' => $age,
        ]);
    }
}
```

Pour tester, allez à `localhost:8000/home/5`

Symfony

Solution avec les attributs

```
namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\Routing\Annotation\Route;

class HomeController extends AbstractController
{
    #[Route('/home/{age<\d+>}', name: "home_route")]
    public function index(int $age)
    {
        return $this->render('home/index.html.twig', [
            'controller_name' => $age,
        ]);
    }
}
```

Symfony

Solution avec les attributs

```
namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\Routing\Annotation\Route;

class HomeController extends AbstractController
{
    #[Route('/home/{age<\d+>}', name: "home_route")]
    public function index(int $age)
    {
        return $this->render('home/index.html.twig', [
            'controller_name' => $age,
        ]);
    }
}
```

Pour tester, allez à `localhost:8000/home/5`

Symfony

On peut aussi rendre ce paramètre optionnel en lui attribuant une valeur par défaut

```
namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\Routing\Annotation\Route;

class HomeController extends AbstractController
{
    #[Route('/home/{age<\d+>}', name: "home_route")]
    public function index(int $age = 7)
    {
        return $this->render('home/index.html.twig', [
            'controller_name' => $age,
        ]);
    }
}
```

Symfony

On peut aussi rendre ce paramètre optionnel en lui attribuant une valeur par défaut

```
namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\Routing\Annotation\Route;

class HomeController extends AbstractController
{
    #[Route('/home/{age<\d+>}', name: "home_route")]
    public function index(int $age = 7)
    {
        return $this->render('home/index.html.twig', [
            'controller_name' => $age,
        ]);
    }
}
```

Pour tester, allez à `localhost:8000/home`

Symfony

Pour accepter la valeur `null`

```
namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\Routing\Annotation\Route;

class HomeController extends AbstractController
{
    #[Route('/home/{age?}', name: "home_route")]
    public function index(?int $age)
    {
        return $this->render('home/index.html.twig', [
            'controller_name' => $age,
        ]);
    }
}
```

Symfony

Pour accepter la valeur `null`

```
namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\Routing\Annotation\Route;

class HomeController extends AbstractController
{
    #[Route('/home/{age?}', name: "home_route")]
    public function index(?int $age)
    {
        return $this->render('home/index.html.twig', [
            'controller_name' => $age,
        ]);
    }
}
```

Pour tester, allez à `localhost:8000/home`

On peut aussi définir des constantes qu'on récupère comme des paramètres de substitution mais ils ne font pas partis de la route

```
namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\Routing\Annotation\Route;

class HomeController extends AbstractController
{
    /**
     * @Route("/home/{age}", name="home_route", defaults={"nom": "wick", "prenom": "john"})
     */
    public function index(int $age, string $nom, string $prenom)
    {
        return $this->render('home/index.html.twig', [
            'controller_name' => "$age $nom $prenom",
        ]);
    }
}
```

On peut aussi définir des constantes qu'on récupère comme des paramètres de substitution mais ils ne font pas partis de la route

```
namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\Routing\Annotation\Route;

class HomeController extends AbstractController
{
    /**
     * @Route("/home/{age}", name="home_route", defaults={"nom": "wick", "prenom": "john"})
     */
    public function index(int $age, string $nom, string $prenom)
    {
        return $this->render('home/index.html.twig', [
            'controller_name' => "$age $nom $prenom",
        ]);
    }
}
```

Pour tester, allez à localhost:8000/home/45

Symfony

Solution avec les attributs

```
namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\Routing\Annotation\Route;

class HomeController extends AbstractController
{
    #[Route('/home/{age}', name: "home_route", defaults: ["nom" => "
        wick", "prenom" => "john"])]
    public function index(int $age, string $nom, string $prenom)
    {
        return $this->render('home/index.html.twig', [
            'controller_name' => "$age $nom $prenom",
        ]);
    }
}
```

Symfony

Solution avec les attributs

```
namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\Routing\Annotation\Route;

class HomeController extends AbstractController
{
    #[Route('/home/{age}', name: "home_route", defaults: ["nom" => "
        wick", "prenom" => "john"])]
    public function index(int $age, string $nom, string $prenom)
    {
        return $this->render('home/index.html.twig', [
            'controller_name' => "$age $nom $prenom",
        ]);
    }
}
```

Pour tester, allez à localhost:8000/home/45

Symfony

Considérons le contrôleur suivant

```
class HomeController extends AbstractController
{
    #[Route('/home/{nom}', name: 'home_route')]
    public function index(string $nom)
    {
        return $this->render('home/index.html.twig', [
            'controller_name' => $nom,
        ]);
    }

    #[Route('/home/index', name: 'home_route2')]
    public function index2()
    {
        return $this->render('home/index.html.twig', [
            'controller_name' => "HomeController",
        ]);
    }
}
```

Symfony

Constats

- En allant à `localhost:8000/home/wick` \Rightarrow Hello wick! s'affiche.
- En allant à `localhost:8000/home/index` \Rightarrow Hello index! s'affiche.

© Achret

Symfony

Constats

- En allant à `localhost:8000/home/wick` \Rightarrow Hello wick! s'affiche.
- En allant à `localhost:8000/home/index` \Rightarrow Hello index! s'affiche.

Conclusion

L'action `index2` n'est jamais exécutée car la route demandée match avec la route de l'action `index`.

Symfony

Solution

- Depuis **Symfony 5.1**, il est possible d'associer une valeur de priorité à chaque route.
- Une route sans la propriété `priority` a par défaut la priorité 0.
- Si la route demandée match avec plusieurs routes définies, alors celle avec la priorité max sera exécutée.

Ajoutons une priorité à la deuxième route

```
class HomeController extends AbstractController
{
    #[Route('/home/{nom}', name: 'home_route')]
    public function index(string $nom)
    {
        return $this->render('home/index.html.twig', [
            'controller_name' => $nom,
        ]);
    }

    #[Route('/home/index', name: 'home_route2', priority: 2)]
    public function index2()
    {
        return $this->render('home/index.html.twig', [
            'controller_name' => "HomeController",
        ]);
    }
}
```

Symfony

Constats

- En allant à `localhost:8000/home/wick` ⇒ Hello wick! s'affiche.
- En allant à `localhost:8000/home/index` ⇒ Hello HomeController! s'affiche.

© Achref EL MOUELHI

Symfony

Constats

- En allant à `localhost:8000/home/wick` \Rightarrow Hello wick! s'affiche.
- En allant à `localhost:8000/home/index` \Rightarrow Hello HomeController! s'affiche.

Question

Comment on faisait avant **Symfony 5.1** ?

Symfony

Constats

- En allant à `localhost:8000/home/wick` ⇒ Hello wick! s'affiche.
- En allant à `localhost:8000/home/index` ⇒ Hello HomeController! s'affiche.

Question

Comment on faisait avant **Symfony 5.1** ?

Réponse

L'exécution est séquentielle, on jouait sur l'ordre de définition de nos actions (routes).

Symfony

Pour consulter les routes disponibles

```
php bin/console debug:router
```

© Achref EL MOUELHI ©

Symfony

Pour consulter les routes disponibles

```
php bin/console debug:router
```

Le résultat

Name	Method	Scheme	Host	Path
home_route2	ANY	ANY	ANY	/home/index
_preview_error	ANY	ANY	ANY	/_error/{code}.{_format}
_wdt	ANY	ANY	ANY	/_wdt/{token}
_profiler_home	ANY	ANY	ANY	/_profiler/
_profiler_search	ANY	ANY	ANY	/_profiler/search
_profiler_search_bar	ANY	ANY	ANY	/_profiler/search_bar
_profiler_phpinfo	ANY	ANY	ANY	/_profiler/phpinfo
_profiler_search_results	ANY	ANY	ANY	/_profiler/{token}/search/results
_profiler_open_file	ANY	ANY	ANY	/_profiler/open
_profiler	ANY	ANY	ANY	/_profiler/{token}
_profiler_router	ANY	ANY	ANY	/_profiler/{token}/router
_profiler_exception	ANY	ANY	ANY	/_profiler/{token}/exception
_profiler_exception_css	ANY	ANY	ANY	/_profiler/{token}/exception.css
home_route	ANY	ANY	ANY	/home/{nom}

Symfony

On peut aussi spécifier le nom de la route

```
php bin/console debug:router home_route
```

© Achref EL MOUELHI ©

Symfony

On peut aussi spécifier le nom de la route

```
php bin/console debug:router home_route
```

Le résultat

Property	Value
Route Name	home_route
Path	/home/{nom}
Path Regex	{^/home/(?P<nom>[/]++)}\$sDu
Host	ANY
Host Regex	
Scheme	ANY
Method	ANY
Requirements	NO CUSTOM
Class	Symfony\Component\Routing\Route
Defaults	_controller: App\Controller\HomeController::index()
Options	compiler_class: Symfony\Component\Routing\RouteCompiler
	utf8: true

Symfony

Pour tester si une route match avec une route définie (**A ne pas essayer depuis une console Bash**)

```
php bin/console router:match /home/john
```

© Achref EL MOUELHI ©

Symfony

Pour tester si une route match avec une route définie (**A ne pas essayer depuis une console Bash**)

```
php bin/console router:match /home/john
```

Le résultat

[OK] Route "home_route" matches

Property	Value
Route Name	home_route
Path	/home/{nom}
Path Regex	{^/home/(?P<nom>[^\/]++)\$}sDu
Host	ANY
Host Regex	
Scheme	ANY
Method	ANY
Requirements	NO CUSTOM
Class	Symfony\Component\Routing\Route
Defaults	_controller: App\Controller\HomeController::index()
Options	compiler_class: Symfony\Component\Routing\RouteCompiler
	utf8: true

L'objet `request` permet de

- récupérer les paramètres de substitution
- récupérer les variables hors routes (les paramètres libres)
- récupérer la méthode de la requête **HTTP**
- récupérer le nom de la route
- ...

Pour récupérer un paramètre de substitution en utilisant l'objet `request`

```
namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\Routing\Annotation\Route;
use Symfony\Component\HttpFoundation\Request;

class HomeController extends AbstractController
{
    #[Route('/home/{nom}', name: "home_route")]
    public function index(Request $request)
    {
        $nom = $request->attributes->get('nom');
        return $this->render('home/index.html.twig', [
            'controller_name' => $nom,
        ]);
    }
}
```

Pour récupérer un paramètre de substitution en utilisant l'objet `request`

```
namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\Routing\Annotation\Route;
use Symfony\Component\HttpFoundation\Request;

class HomeController extends AbstractController
{
    #[Route('/home/{nom}', name: "home_route")]
    public function index(Request $request)
    {
        $nom = $request->attributes->get('nom');
        return $this->render('home/index.html.twig', [
            'controller_name' => $nom,
        ]);
    }
}
```

Pour tester, allez à `localhost:8000/home/wick`

On peut aussi utiliser le raccourci `request->get('nom_parametre')`

```
namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\Routing\Annotation\Route;
use Symfony\Component\HttpFoundation\Request;

class HomeController extends AbstractController
{
    #[Route('/home/{nom}', name: "home_route")]
    public function index(Request $request)
    {
        $nom = $request->get('nom');
        return $this->render('home/index.html.twig', [
            'controller_name' => $nom,
        ]);
    }
}
```


On peut aussi utiliser le raccourci `request->get('nom_parametre')`

```
namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\Routing\Annotation\Route;
use Symfony\Component\HttpFoundation\Request;

class HomeController extends AbstractController
{
    #[Route('/home/{nom}', name: "home_route")]
    public function index(Request $request)
    {
        $nom = $request->get('nom');
        return $this->render('home/index.html.twig', [
            'controller_name' => $nom,
        ]);
    }
}
```

Pour tester, allez à `localhost:8000/home/wick`

On peut aussi utiliser le raccourci `attributes` pour récupérer tous les paramètres

```
namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\Routing\Annotation\Route;
use Symfony\Component\HttpFoundation\Request;

class HomeController extends AbstractController
{
    #[Route('/home/{nom}/{prenom}', name: "home_route")]
    public function index(Request $request)
    {
        $params = $request->attributes->get('_route_params');
        return $this->render('home/index.html.twig', [
            'controller_name' => implode(" ", $params),
        ]);
    }
}
```

On peut aussi utiliser le raccourci `attributes` pour récupérer tous les paramètres

```
namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\Routing\Annotation\Route;
use Symfony\Component\HttpFoundation\Request;

class HomeController extends AbstractController
{
    #[Route('/home/{nom}/{prenom}', name: "home_route")]
    public function index(Request $request)
    {
        $params = $request->attributes->get('_route_params');
        return $this->render('home/index.html.twig', [
            'controller_name' => implode(" ", $params),
        ]);
    }
}
```

Pour tester, allez à `localhost:8000/home/wick/john`

Pour les paramètres hors routes, pas besoin de les déclarer dans la route

```
namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\Routing\Annotation\Route;
use Symfony\Component\HttpFoundation\Request;

class HomeController extends AbstractController
{
    #[Route('/home', name: "home_route")]
    public function index(Request $request)
    {
        $nom = $request->query->get('nom');
        return $this->render('home/index.html.twig', [
            'controller_name' => $nom,
        ]);
    }
}
```

Pour les paramètres hors routes, pas besoin de les déclarer dans la route

```
namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\Routing\Annotation\Route;
use Symfony\Component\HttpFoundation\Request;

class HomeController extends AbstractController
{
    #[Route('/home', name: "home_route")]
    public function index(Request $request)
    {
        $nom = $request->query->get('nom');
        return $this->render('home/index.html.twig', [
            'controller_name' => $nom,
        ]);
    }
}
```

Pour tester, allez à `localhost:8000/home?nom=wick`

On peut aussi utiliser le raccourci `request->get('nom_parametre')`

```
namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\Routing\Annotation\Route;
use Symfony\Component\HttpFoundation\Request;

class HomeController extends AbstractController
{
    #[Route('/home', name: "home_route")]
    public function index(Request $request)
    {
        $nom = $request->get('nom');
        return $this->render('home/index.html.twig', [
            'controller_name' => $nom,
        ]);
    }
}
```

On peut aussi utiliser le raccourci `request->get('nom_parametre')`

```
namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\Routing\Annotation\Route;
use Symfony\Component\HttpFoundation\Request;

class HomeController extends AbstractController
{
    #[Route('/home', name: "home_route")]
    public function index(Request $request)
    {
        $nom = $request->get('nom');
        return $this->render('home/index.html.twig', [
            'controller_name' => $nom,
        ]);
    }
}
```

Pour tester, allez à `localhost:8000/home?nom=wick`

Symfony

Pour récupérer le nom d'une route surtout en cas de multi-routing

```
namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\Routing\Annotation\Route;
use Symfony\Component\HttpFoundation\Request;

class HomeController extends AbstractController
{
    #[Route('/index', name: "index_route")]
    #[Route('/home', name: "home_route")]
    public function index(Request $request)
    {
        $route = $request->attributes->get('_route');
        return $this->render('home/index.html.twig', [
            'controller_name' => $route,
        ]);
    }
}
```


Symfony

Constats

- En allant à `localhost:8000/home` \Rightarrow Hello `home_route!` s'affiche.
- En allant à `localhost:8000/index` \Rightarrow Hello `index_route!` s'affiche.

Symfony

Autres informations contenues dans `$request`

- `$request->server` : les variables de serveur
- `$request->cookies` : les variables de cookie
- `$request->getMethod()` : le nom de la méthode **HTTP**
- `$request->attributes->get('_route')` ou son raccourci `$request->get('_route')` : le nom de la route
- `$request->request->get('var')` : une variable envoyée par le biais de la méthode `POST`
- ...

Remarque

- Par défaut, chaque action d'un contrôleur peut être exécutée quelle que soit la méthode **HTTP**.
- Cependant, il est possible de spécifier pour chaque action les méthodes **HTTP** autorisées.

Symfony

L'action `index` sera exécutée pour les deux méthodes HTTP `GET` et `POST`.

```
namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\Routing\Annotation\Route;

class HomeController extends AbstractController
{
    /**
     * @Route("/home", name="home_route", methods={"GET", "POST"})
     */
    public function index(Request $request)
    {
        $route = $request->getMethod();
        return $this->render('home/index.html.twig', [
            'controller_name' => $route,
        ]);
    }
}
```

Symfony

Solution avec les attributs

```
namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\Routing\Annotation\Route;
use Symfony\Component\HttpFoundation\Request;

class HomeController extends AbstractController
{
    #[Route('/home', name: "home_route", methods: ["GET", "POST"])]
    public function index(Request $request)
    {
        $route = $request->getMethod();
        return $this->render('home/index.html.twig', [
            'controller_name' => $route,
        ]);
    }
}
```

Symfony

Constat

En allant à `localhost:8000/home` \Rightarrow Hello GET! s'affiche.

Symfony

Expression Language

- Composant **Symfony** introduit dans la version 2.4
- Inspiré par **SpEL** (Spring Expression Language)
- Basé sur la syntaxe de **Twig**
- Permettant de simplifier les contrôles

Symfony

Exemple

```
namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\Routing\Annotation\Route;
use Symfony\Component\HttpFoundation\Request;

class HomeController extends AbstractController
{
    /**
     * @Route("/home", name="home_route", condition="context.getMethod
        () in ['GET', 'POST'])
     */
    public function index(Request $request)
    {
        $methode = $request->getMethod();
        return $this->render('home/index.html.twig', [
            'controller_name' => $methode,
        ]);
    }
}
```


Symfony

Explication

- L'action `index` sera exécutée pour les deux méthodes **HTTP** `GET` et `POST`.
- `context` : instance de `Symfony\Component\Routing\RequestContext`.

© Achref EL MOUËZ

Symfony

Explication

- L'action `index` sera exécutée pour les deux méthodes **HTTP** `GET` et `POST`.
- `context` : instance de `Symfony\Component\Routing\RequestContext`.

Autres variables disponibles dans EL

- `request` : instance de `Symfony\Component\HttpFoundation\Request`.
- `this` : pour accéder aux attributs de l'objet courant.
- ...

Symfony

Opérateurs arithmétiques supportés

- +
- -
- *
- /
-
- ** (puissance)

Symfony

Opérateurs de comparaison

- ==
- ===
- !=
- !==
- <
- >
- <=
- >=
- matches

Symfony

Opérateurs logiques

- not **ou** !
- and **ou** &&
- or **ou** ||

© Achrel

Symfony

Opérateurs logiques

- not **ou** !
- and **ou** &&
- or **ou** ||

Opérateurs de concaténation (string)

~

Symfony

Opérateurs pour les tableaux

- `in`
- `not in`
- `..`

© Achref EL MOU

Symfony

Opérateurs pour les tableaux

- `in`
- `not in`
- `..`

Opérateurs ternaires

- `condition ? vrai : faux`
- `condition ? : faux`
- `condition ? vrai`

Symfony

Commençons par générer un deuxième contrôleur nommé
VehiculeController

```
php bin/console make:controller VehiculeController
```

© Achref EL MOUL

Symfony

Commençons par générer un deuxième contrôleur nommé
`VehiculeController`

```
php bin/console make:controller VehiculeController
```

Le résultat est

```
created: src/Controller/VehiculeController.php  
created: templates/vehicule/index.html.twig
```

Symfony

Considérons le contenu suivant pour `HomeController`

```
namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\Routing\Annotation\Route;

class HomeController extends AbstractController
{
    #[Route('/home/{nom}', name: "home_route")]
    public function index(string $nom)
    {
        return $this->render('home/index.html.twig', [
            'controller_name' => $nom,
        ]);
    }
}
```

Symfony

Objectif

Dans `VehiculeController` : **générer une route et rediriger vers l'action `index` de `HomeController`**

Symfony

Une première solution consiste à utiliser la méthode `forward`

```
namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Routing\Annotation\Route;

class VehiculeController extends AbstractController
{
    #[Route('/vehicule', name: 'vehicule_route')]
    public function index(): Response
    {
        return $this->forward('App\Controller\HomeController::index', [
            'nom' => 'abruzzi',
        ]);
    }
}
```

Symfony

Une première solution consiste à utiliser la méthode `forward`

```
namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Routing\Annotation\Route;

class VehiculeController extends AbstractController
{
    #[Route('/vehicule', name: 'vehicule_route')]
    public function index(): Response
    {
        return $this->forward('App\Controller\HomeController::index', [
            'nom' => 'abruzzi',
        ]);
    }
}
```

Pour tester, allez à `localhost:8000/vehicule`

Symfony

Constats

La route affichée dans la barre d'adresse (`/vehicule`) ne correspond pas au contenu affiché (appartenant au contrôleur `HomeController`).

Dans `VehiculeController`, on génère une URL puis on redirige

```
namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\HttpFoundation\RedirectResponse;
use Symfony\Component\Routing\Annotation\Route;

class VehiculeController extends AbstractController
{
    #[Route('/vehicule', name: 'vehicule_route')]
    public function index(): Response
    {
        $url = $this->generateUrl('home_route', array(
            'nom' => 'abruzzi',
        ));
        return new RedirectResponse($url);
    }
}
```


Dans `VehiculeController`, on génère une URL puis on redirige

```
namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\HttpFoundation\RedirectResponse;
use Symfony\Component\Routing\Annotation\Route;

class VehiculeController extends AbstractController
{
    #[Route('/vehicule', name: 'vehicule_route')]
    public function index(): Response
    {
        $url = $this->generateUrl('home_route', array(
            'nom' => 'abruzzi',
        ));
        return new RedirectResponse($url);
    }
}
```

Pour tester, allez à `localhost:8000/vehicule`

On peut aussi utiliser la méthode `redirect` de `AbstractController`

```
namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Routing\Annotation\Route;

class VehiculeController extends AbstractController
{
    #[Route('/vehicule', name: 'vehicule_route')]
    public function index(): Response
    {
        $url = $this->generateUrl('home_route', array(
            'nom' => 'abruzzi',
        ));
        return $this->redirect($url);
    }
}
```

On peut aussi utiliser la méthode `redirect` de `AbstractController`

```
namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Routing\Annotation\Route;

class VehiculeController extends AbstractController
{
    #[Route('/vehicule', name: 'vehicule_route')]
    public function index(): Response
    {
        $url = $this->generateUrl('home_route', array(
            'nom' => 'abruzzi',
        ));
        return $this->redirect($url);
    }
}
```

Pour tester, allez à `localhost:8000/vehicule`

Symfony

On peut aussi utiliser le raccourci `redirectToRoute`

```
namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Routing\Annotation\Route;

class VehiculeController extends AbstractController
{
    #[Route('/vehicule', name: 'vehicule_route')]
    public function index(): Response
    {
        return $this->redirectToRoute('home_route', [
            'nom' => 'abruzzi'
        ]);
    }
}
```

Symfony

On peut aussi utiliser le raccourci `redirectToRoute`

```
namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Routing\Annotation\Route;

class VehiculeController extends AbstractController
{
    #[Route('/vehicule', name: 'vehicule_route')]
    public function index(): Response
    {
        return $this->redirectToRoute('home_route', [
            'nom' => 'abruzzi'
        ]);
    }
}
```

Pour tester, allez à `localhost:8000/vehicule`

Symfony

La méthode `redirectToRoute` permet aussi de rediriger vers une URL externe

```
namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\Routing\Annotation\Route;

class VehiculeController extends AbstractController
{
    #[Route('/vehicule', name: 'vehicule_route')]
    public function index()
    {
        return $this->redirectToRoute('http://symfony.com/doc');
    }
}
```

Symfony

La méthode `redirectToRoute` permet aussi de rediriger vers une URL externe

```
namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\Routing\Annotation\Route;

class VehiculeController extends AbstractController
{
    #[Route('/vehicule', name: 'vehicule_route')]
    public function index()
    {
        return $this->redirectToRoute('http://symfony.com/doc');
    }
}
```

Pour tester, allez à `localhost:8000/vehicule`

Pour renvoyer une page d'erreur, on peut utiliser `HttpException`

```
namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Routing\Annotation\Route;
use Symfony\Component\HttpKernel\Exception\HttpException;

class HomeController extends AbstractController
{
    #[Route('/home/{nom?}', name: "home_route")]
    public function index(?string $nom): Response
    {
        if (!isset($nom)) {
            throw new HttpException(
                404,
                'On ne peut vous afficher la page de cette personne'
            );
        }
        return $this->render('home/index.html.twig', [
            'controller_name' => $nom,
        ]);
    }
}
```


Pour tester

- allez à une URL avec paramètre `localhost:8000/home/wick`
- et une deuxième sans `localhost:8000/home`

Symfony

On peut aussi utiliser la méthode `createNotFoundException`

```
namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Routing\Annotation\Route;

class HomeController extends AbstractController
{
    #[Route('/home/{nom?}', name: "home_route")]
    public function index(?string $nom): Response
    {
        if (!isset($nom)) {
            throw $this->createNotFoundException('On ne peut vous
                afficher la page de cette personne');
        }
        return $this->render('home/index.html.twig', [
            'controller_name' => $nom,
        ]);
    }
}
```

Symfony

Deux utilisations possibles pour l'objet `response`

- explicite : en construisant la réponse
- implicite : on n'utilise pas l'objet `response` pour retourner la réponse mais il sera utilisé en coulisses, nous n'avons pas à le manipuler directement.

Utilisation explicite

```
namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\Routing\Annotation\Route;
use Symfony\Component\HttpFoundation\Response;

class HomeController extends AbstractController
{
    #[Route('/home/{nom?}', name: "home_route")]
    public function index(?string $nom): Response
    {
        $response = new Response(
            "<p>Bonjour $nom</p>",
            Response::HTTP_OK,
            ['content-type' => 'text/html']
        );
        return $response;
    }
}
```

Utilisation explicite

```
namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\Routing\Annotation\Route;
use Symfony\Component\HttpFoundation\Response;

class HomeController extends AbstractController
{
    #[Route('/home/{nom?}', name: "home_route")]
    public function index(?string $nom): Response
    {
        $response = new Response(
            "<p>Bonjour $nom</p>",
            Response::HTTP_OK,
            ['content-type' => 'text/html']
        );
        return $response;
    }
}
```

Pour tester, allez à `localhost:8000/home/wick`

Symfony

Utilisation implicite

```
namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Routing\Annotation\Route;

class HomeController extends AbstractController
{
    #[Route('/home/{nom?}', name: "home_route")]
    public function index(?string $nom): Response
    {
        return $this->render('home/index.html.twig', [
            'controller_name' => $nom,
        ]);
    }
}
```

Deux types de paramètre d'application

- prédéfinis : quelle que soit l'application **Symfony**
- personnalisés : à définir par le développeur

Pour récupérer le chemin absolu vers notre projet

```
namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Routing\Annotation\Route;

class HomeController extends AbstractController
{
    #[Route('/home/{nom?}', name: "home_route")]
    public function index(?string $nom): Response
    {
        $path = $this->getParameter('kernel.project_dir');
        return $this->render('home/index.html.twig', [
            'controller_name' => $path,
        ]);
    }
}
```


Pour récupérer le chemin absolu vers notre projet

```
namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Routing\Annotation\Route;

class HomeController extends AbstractController
{
    #[Route('/home/{nom?}', name: "home_route")]
    public function index(?string $nom): Response
    {
        $path = $this->getParameter('kernel.project_dir');
        return $this->render('home/index.html.twig', [
            'controller_name' => $path,
        ]);
    }
}
```

En allant localhost:8000/home/wick ⇒ Hello
C:\Users\elmou\Desktop\premier_projet_symfony! est affiché

Autres paramètres définis

- `kernel.charset` pour l'encodage.
- `kernel.cache_dir` pour le chemin vers le dossier du cache
(C:\Users\elmou\Desktop\premier-projet-symfony\var\cache\dev).
- `kernel.logs_dir` pour le chemin vers le dossier de journalisation
(C:\Users\elmou\Desktop\premier-projet-symfony\var\log).
- ...

Paramètres personnalisés

- Pour définir nos paramètres, il faut aller dans `config/services.yaml` et ajouter le nouveau paramètre dans la section `parameters` sous format : clé : valeur.
- Ensuite, on utilise la méthode `$this->getParameter()` pour récupérer le paramètre.

Symfony

Commençons par définir le paramètre suivant dans la section
`parameters` **de** `config/services.yaml`

```
# This file is the entry point to configure your own
services.

# Files in the packages/ subdirectory configure your
dependencies.

# Put parameters here that don't need to change on
each machine where the app is deployed
# https://symfony.com/doc/current/best\_practices/
configuration.html#application-related-
configuration

parameters:
    nom: 'wick'
```

Pour récupérer le paramètre personnalisé

```
namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Routing\Annotation\Route;

class HomeController extends AbstractController
{
    #[Route('/home', name: "home_route")]
    public function index(): Response
    {
        $nom = $this->getParameter('nom');
        return $this->render('home/index.html.twig', [
            'controller_name' => $nom,
        ]);
    }
}
```

Pour récupérer le paramètre personnalisé

```
namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Routing\Annotation\Route;

class HomeController extends AbstractController
{
    #[Route('/home', name: "home_route")]
    public function index(): Response
    {
        $nom = $this->getParameter('nom');
        return $this->render('home/index.html.twig', [
            'controller_name' => $nom,
        ]);
    }
}
```

En allant localhost:8000/home ⇒ Hello wick! est affiché.

Bonnes pratiques

- Faire hériter son contrôleur de `AbstractController`.
- Utiliser plutôt les attributs/annotations pour la configuration : routage, sécurité...
- Favoriser l'injection de dépendance pour l'utilisation des services.
- Ne plus organiser l'application par **Bundle** et utiliser plutôt les namespaces. (Les bundles ont été utilisés dans les premières versions de **Symfony** pour créer des fragments réutilisables).